МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА"

СОГЛАСОВАНО
Зав. выпускающей кафедры

УТВЕРЖДАЮ

Архитектуры промышленных программных систем

рабочая программа дисциплины (модуля)

Закреплена за кафедрой	**Систем автоматизированного проектирования вычислительных средств**
Учебный план	Лицензирование_02.04.02_25_00.plx 02.04.02 Фундаментальная информатика и информационные технологии
Квалификация	**магистр**
Форма обучения	**очная**
Общая трудоемкость	**5 ЗЕТ**

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**1 (1.1)**		Итого	
Недель	16			
Вид занятий	УП	РП	УП	РП
Лекции	16	16	16	16
Практические	16	16	16	16
Иная контактная работа	0,35	0,35	0,35	0,35
Консультирование перед экзаменом и практикой	2	2	2	2
Итого ауд.	34,35	34,35	34,35	34,35
Контактная работа	34,35	34,35	34,35	34,35
Сам. работа	101		101	
Часы на контроль	44,65		44,65	
Итого	180	34,35	180	34,35

г. Рязань

Программу составил(и):
к.т.н, доцент, Бакулев Александр Валериевич

Рабочая программа дисциплины
Архитектуры промышленных программных систем

разработана в соответствии с ФГОС ВО:
ФГОС ВО - магистратура по направлению подготовки 02.04.02 Фундаментальная информатика и информационные технологии (приказ Минобрнауки России от 23.08.2017 г. № 811)

составлена на основании учебного плана:
02.04.02 Фундаментальная информатика и информационные технологии
утвержденного учёным советом вуза от 25.04.2025 протокол № 12.

Рабочая программа одобрена на заседании кафедры
Систем автоматизированного проектирования вычислительных средств

Протокол от 04.07.2025 г. № 8
Срок действия программы: уч.г.
Зав. кафедрой Корячко Вячеслав Петрович

Визирование РПД для исполнения в очередном учебном году

Рабочая программа пересмотрена, обсуждена и одобрена для
исполнения в 2026-2027 учебном году на заседании кафедры
Систем автоматизированного проектирования вычислительных средств

Протокол от __ _____ 2026 г. № __

Зав. кафедрой _____

Визирование РПД для исполнения в очередном учебном году

Рабочая программа пересмотрена, обсуждена и одобрена для
исполнения в 2027-2028 учебном году на заседании кафедры
Систем автоматизированного проектирования вычислительных средств

Протокол от __ _____ 2027 г. № __

Зав. кафедрой _____

Визирование РПД для исполнения в очередном учебном году

Рабочая программа пересмотрена, обсуждена и одобрена для
исполнения в 2028-2029 учебном году на заседании кафедры
Систем автоматизированного проектирования вычислительных средств

Протокол от __ _____ 2028 г. № __

Зав. кафедрой _____

Визирование РПД для исполнения в очередном учебном году

Рабочая программа пересмотрена, обсуждена и одобрена для
исполнения в 2029-2030 учебном году на заседании кафедры

Систем автоматизированного проектирования вычислительных средств

Протокол от __ _____ 2029 г. № __

Зав. кафедрой _____

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)	
1.1	Одной из ключевых проблем, возникающих при разработке современных информационных систем промышленного (Enterprise) уровня является проектирование или выбор подходящей для построения будущей сложной системы архитектуры. Цель данной дисциплины - систематизация знаний об архитектурах промышленных программных систем и принципах их проектирования.

2. МЕСТО ДИСЦИПЛИНЫ (МОДУЛЯ) В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Цикл (раздел) ОП:	Б1.В
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Для успешного освоения данной дисциплины требуются знания языков программирования, общих принципов объектно-ориентированного проектирования программ, навыки использования инструментальных средств разработки программного обеспечения.
2.2	**Дисциплины (модули) и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Объектный анализ и объектно-ориентированное программирование
2.2.2	Методологии разработки программного обеспечения
2.2.3	Инструментальные средства разработки программного обеспечения
2.2.4	Разработка системного программного обеспечения
2.2.5	Производственная практика
2.2.6	Научно-исследовательская работа (концентрированная)
2.2.7	Преддипломная практика
2.2.8	Производственная практика
2.2.9	Эксплуатационная практика
2.2.10	Подготовка к процедуре защиты и защита выпускной квалификационной работы
2.2.11	Продуктовая аналитика
2.2.12	Системный анализ
2.2.13	Облачные технологии

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

УК-2: Способен управлять проектом на всех этапах его жизненного цикла

УК-2.1. Осуществляет управление проектом на всех этапах жизненного цикла

Знать
Жизненный цикл проекта и его этапы: инициация, планирование, реализация, мониторинг/контроль, завершение.
Методологии управления проектами: Agile, Waterfall, Scrum, Kanban.
Уметь
Определять цели и задачи проекта, формулировать цели, KPI, требования.

Владеть
Инструменты управления проектами, использовать Jira, MS Project.

УК-2.2. Осущевляет обоснованный выбор применяемых программных средств и решений при реализации проекта

Знать
Типы программных решений, технологии разработки ПО.
Уметь
Формулировать требования к ПО.
Анализировать рынок решений.

Владеть
Пользоваться инструментами анализа и выбора ПО и принимать обоснованные решения по их выбору.

ПК-2: Способен управлять архитектурой единой информационной среды

ПК-2.1. Выполняет выбор, согласование требований и моделирование архитектуры единой информационной среды

Знать Требования к информационной системе (бизнес-цели, функционал, безопасность). Архитектурные подходы. **Уметь** Формулировать требования к ЕИС. Моделировать структуру и процессы системы. **Владеть** Инструментами моделирования. Навыками анализа и проектирования систем.
ПК-2.2. Выполняет контроль проектирования и документирования программного обеспечения и его интеграции с точки зрения единой информационной среды
Знать Процессы проектирования и документирования ПО. Требования к взаимодействию систем и обмену данными. **Уметь** Контролировать соответствие проекта требованиям. Проверять корректность архитектурных решений. **Владеть** Инструментами контроля версий. Методиками проверки и тестирования ПО.

ПК-4: Способен осуществлять организацию процессов разработки компьютерного программного обеспечения
ПК-4.1. Выполняет управление процессами проектирования и разработки компьютерного программного обеспечения
Знать Жизненный цикл ПО. Методы проектирования и моделирования систем. **Уметь** Организовать процессы проектирования и разработки. Управлять задачами и ресурсами команды. **Владеть** Навыками координации работы команды. Техниками оценки и планирования работ.
ПК-4.2. Выполняет управление конфигурациями и выпусками программного продукта
Знать Принципы управления конфигурациями. Жизненный цикл релиза ПО. Стандарты CI/CD. **Уметь** Управлять версиями исходного кода и зависимостей. Организовать процессы сборки, тестирования и деплоя. Контролировать изменения в конфигурации ПО. **Владеть** Инструментами CI/CD. Системами контроля версий. Навыками написания скриптов и автоматизации задач.

В результате освоения дисциплины (модуля) обучающийся должен

3.1	**Знать:**
3.2	**Уметь:**
3.3	**Владеть:**

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Компетен-ции	Литература	Форма контроля
	Раздел 1. Основы архитектуры промышленных программных систем.					
1.1	Введение. Основные понятия. /Тема/	1	0			
1.2	/Лек/	1	1	УК-2.2-У ПК-2.1-З		
1.3	Архитектурные шаблоны промышленных программных систем. /Тема/	1	0			

1.4	/Лек/	1	1	УК-2.2-З ПК-2.1-З ПК-2.2-У ПК-4.1-У	Л1.1	
1.5	/ИКР/	1	0,35			
1.6	Системы версионного контроля /Тема/	1	0			
1.7	/Пр/	1	2	ПК-2.2-В ПК-4.1-В ПК-4.2-В	Л1.6Л3.1	
	Раздел 2. Принципы проектирования архитектур программных систем.					
2.1	Системный подход к проектированию программной архитектуры. /Тема/	1	0			
2.2	/Лек/	1	1	УК-2.1-З ПК-2.1-З ПК-2.1-В	Л1.5	
2.3	Архитектурные паттерны SOLID. /Тема/	1	0			
2.4	/Лек/	1	1	УК-2.2-З ПК-4.1-У	Л1.8	
2.5	/Пр/	1	2		Э1 Э2	
	Раздел 3. Классические архитектуры программных систем.					
3.1	Многозвенная архитектура, толстый и тонкий клиенты. /Тема/	1	0			
3.2	/Лек/	1	1	УК-2.1-З УК-2.2-З ПК-2.1-З	Л1.8	
3.3	/Пр/	1	2	ПК-2.2-У ПК-4.1-З ПК-4.1-В ПК-4.2-З	Л2.1	
	Раздел 4. Современные архитектуры программных систем.					
4.1	Сервис-ориентированная архитектура программных систем, микросервисы. /Тема/	1	0			
4.2	/Лек/	1	2	УК-2.1-З УК-2.2-З ПК-2.1-У ПК-2.1-В	Л1.7	
4.3	/Пр/	1	2	УК-2.2-З ПК-2.2-В ПК-4.1-В ПК-4.2-З		
4.4	Облачные технологии, контейнеризация программных приложений. /Тема/	1	0			
4.5	/Лек/	1	2	УК-2.1-З УК-2.2-В ПК-2.2-З	Л1.3	
4.6	/Пр/	1	2	ПК-2.2-В ПК-4.1-З ПК-4.1-В		
4.7	Архитектуры распределённой потоковой обработки. /Тема/	1	0			
4.8	/Лек/	1	2	УК-2.1-З УК-2.1-В ПК-2.2-У ПК-4.1-У	Л1.2	
4.9	/Пр/	1	2	ПК-2.2-В ПК-4.1-В ПК-4.2-В		

	Раздел 5. Интеграция корпоративных приложений.					
5.1	Задачи и проблемы интеграции, стили интеграции. /Тема/	1	0			
5.2	/Лек/	1	1	УК-2.2-З УК-2.2-В ПК-4.1-З	Л1.5	
5.3	Системы обмена на основе очередей сообщений. /Тема/	1	0			
5.4	/Лек/	1	1	УК-2.1-З ПК-4.1-У ПК-4.2-В	Л1.5	
5.5	Брокер распределённой потоковой обработки сообщений. /Тема/	1	0			
5.6	/Лек/	1	1	ПК-2.2-У ПК-4.1-В ПК-4.2-У	Л1.4	
5.7	/Пр/	1	2	УК-2.2-З ПК-2.2-В ПК-4.1-В		
5.8	Корпоративная интеграционная сервисная шина, паттерны интеграции. /Тема/	1	0			
5.9	/Лек/	1	1	УК-2.2-З УК-2.2-У ПК-2.1-У ПК-2.2-У	Л1.4 Л1.5	
5.10	/Пр/	1	2	УК-2.1-У ПК-2.2-З ПК-4.2-У		
	Раздел 6. Качество программных архитектур.					
6.1	Критерии качества архитектур промышленных программных систем. /Тема/	1	0			
6.2	/Лек/	1	1	УК-2.1-У ПК-2.2-З ПК-2.2-У ПК-4.1-З ПК-4.1-У	Л1.5	
6.3	Консультирование перед экзаменом и практикой. /Тема/	1	0			
6.4	/Кнс/	1	2			

5. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Компетенция "УК-2.1: Осуществляет управление проектом на всех этапах жизненного цикла" подразумевает, что специалист должен быть способен организовать, планировать, контролировать и завершать проекты в соответствии с установленными целями, сроками, ресурсами и качественными требованиями.

Компетенция "УК-2.2: Осуществляет обоснованный выбор применяемых программных средств и решений при реализации проекта" предполагает способность специалиста анализировать требования проекта, оценивать доступные технологии и инструменты, и принимать обоснованные решения по их выбору.

Компетенция "ПК-2.1: Выполняет выбор, согласование требований и моделирование архитектуры единой информационной среды" предполагает способность специалиста участвовать в формировании общей IT-инфраструктуры организации.

Компетенция "ПК-2.2: Выполняет контроль проектирования и документирования программного обеспечения и его интеграции с точки зрения единой информационной среды" направлена на обеспечение целостности и согласованности ПО в рамках общей ИТ-инфраструктуры.

Компетенция "ПК-4.1: Выполняет управление процессами проектирования и разработки компьютерного программного обеспечения" направлена на организацию и контроль всех этапов жизненного цикла ПО.

Компетенция "ПК-4.2: Выполняет управление конфигурациями и выпусками программного продукта" направлена на обеспечение стабильности, контролируемости и воспроизводимости процессов сборки, тестирования и релиза ПО.

Контрольные вопросы по дисциплине:
==========
Что такое архитектурный шаблон?
В чём разница между архитектурным паттерном, дизайнерским паттерном и методологией?
Какие цели преследуются при выборе архитектурного шаблона?
Назовите основные требования к архитектуре ПО.
Как влияет выбор архитектурного шаблона на масштабируемость, поддерживаемость и производительность системы?
Какие существуют основные классы архитектурных шаблонов? Приведите примеры.
Чем отличается монолитная архитектура от микросервисной?
В чём суть слоистой архитектуры (Layered Architecture)?
Какие преимущества и недостатки имеет клиент-серверная архитектура?
Что такое архитектура типа Pipe-Filter? Когда она применяется?
Что такое Model-View-Controller (MVC)? Где он используется?
Как устроена архитектура Model-View-Presenter (MVP)? Чем она отличается от MVC?
Что такое Event-Driven Architecture? Приведите примеры использования.
Как работает архитектура Microservices? Какие у неё основные компоненты?
Что такое Service-Oriented Architecture (SOA)? Как её отличить от микросервисов?
Что такое Domain-Driven Design (DDD)?
Какие элементы входят в DDD-подход: домен, сущности, агрегаты, репозитории?
Как DDD взаимодействует с архитектурными шаблонами?
Что такое Hexagonal Architecture (порт и адаптеры)? Какие плюсы и минусы у этого подхода?
Что такое Clean Architecture (Uncle Bob)? Какие слои в ней выделяются?
Что такое CQRS? В каких случаях его применяют?
Что такое Saga Pattern и как он используется в микросервисах?
Какие есть подходы к обработке транзакций в распределённых системах?
Что такое Event Sourcing? Как оно сочетается с CQRS?
Какие шаблоны используются для обеспечения отказоустойчивости в распределённых системах?
Как определить, какой архитектурный шаблон подходит для конкретного проекта?
Какие факторы влияют на выбор архитектурного шаблона?
Какие ошибки чаще всего возникают при проектировании архитектуры?
Как протестировать соответствие реализации выбранному архитектурному шаблону?
Как документировать архитектуру системы?
Что такое bounded context в контексте DDD?
Какие инструменты используются для моделирования архитектуры?
Что такое hexagonal architecture и как она обеспечивает тестирование?
Какие есть шаблоны для управления состоянием в распределённых системах?
Как работают event brokers и message queues в архитектуре?
Основы версионного контроля
Что такое система версионного контроля (VCS)?
В чём разница между локальным и распределённым VCS?
Какие основные задачи решает система версионного контроля?
Что такое репозиторий?
Перечислите популярные системы версионного контроля.
Что такое команда git init? Для чего она используется?
Как работает команда git add?
Что делает команда git commit?
В чём смысл команды git status?
Что такое ветка (branch) в Git?
Как клонировать удалённый репозиторий?
Что такое origin в Git?
Что означают команды git push и git pull?
Что такое merge и rebase?
В чём разница между git merge и git rebase?
Как создать новую ветку в Git?
Как переключиться между ветками?
Что такое feature branch workflow?
Что такое pull request и зачем он нужен?
Что такое protected branches и где они применяются?
Что такое commit hash?
Как откатить изменения в Git?
Что такое stash и когда его используют?
Что такое cherry-pick?
Как работает git log?
Что такое конфликт слияния?

Как решаются конфликты в Git?
Что такое fast-forward merge?
Что такое squash commit и зачем он нужен?
Как можно объединить несколько коммитов в один?
Что такое detached HEAD?
Что такое tag в Git?
Что такое hook в Git?
Как работает git blame?
Что такое reflog и как он помогает восстановить изменения?
Как Git взаимодействует с CI/CD-системами?
Что такое GitHub Actions, GitLab CI, Jenkins?
Как Git поддерживает автоматическую сборку и тестирование?
Что такое feature toggle и как он связан с Git?
Как Git используется для управления выпусками (releases)?
Что такое submodule в Git?
Что такое fork и clone?
Что такое diff и как его использовать?
Что такое bare repository?
Как Git хранит данные внутри себя?
Что такое SOLID? Перечислите его пять принципов.
Зачем нужны архитектурные паттерны SOLID?
Как SOLID влияет на читаемость и поддерживаемость кода?
В чём разница между архитектурными шаблонами и принципами SOLID?
Почему принципы SOLID особенно важны при работе с крупными системами?
Сформулируйте принцип SRP.
Приведите пример нарушения SRP.
Как SRP связан с модульностью и тестированием?
Можно ли применить SRP к классам и функциям?
Как улучшить архитектуру с помощью SRP?
Сформулируйте принцип OCP.
Что значит быть "открытым для расширения, но закрытым для модификации"?
Приведите пример реализации OCP.
Как использовать интерфейсы или абстракции для соблюдения OCP?
Как OCP помогает в поддержке кода?
Сформулируйте принцип LSP.
Приведите пример нарушения LSP.
Что такое безопасная подстановка?
Как LSP связан с полиморфизмом?
Как проверить, соблюдается ли LSP в коде?
Сформулируйте принцип ISP.
Что такое "толстый" интерфейс и почему он плох?
Приведите пример разбиения интерфейса согласно ISP.
Как ISP влияет на дизайн классов?
Как ISP помогает избежать зависимости от ненужных методов?
Сформулируйте принцип DIP.
Что такое "зависимость от абстракций, а не от деталей"?
Приведите пример реализации DIP.
Как DIP связана с внедрением зависимостей (DI)?
Как DIP помогает в тестировании и модульности?
Какие проблемы может решить применение SOLID-принципов?
Какие ошибки возникают при игнорировании SOLID?
Может ли система работать без соблюдения SOLID? В каких случаях?
Какие паттерны проектирования тесно связаны с SOLID?
Как проверить, соответствует ли ваш проект принципам SOLID?
Как SOLID влияет на CI/CD и автоматизацию?
Как SOLID помогает в рефакторинге кода?
Можно ли применить SOLID в фронтенд-разработке?
Как SOLID работает в микросервисной архитектуре?
Какие инструменты и практики помогают соблюдать SOLID?
Что такое программная архитектура?
В чём разница между архитектурой и проектированием?
Перечислите основные цели выбора архитектуры ПО.
Что такое монолитная архитектура?
Что такое клиент-серверная архитектура?
Как устроена монолитная архитектура? Опишите её компоненты.
Какие преимущества имеет монолитная архитектура?
Какие недостатки характерны для монолита?
В каких случаях применяется монолитная архитектура?

Как монолит влияет на скорость разработки и деплоя?
Что такое клиент и сервер в контексте архитектуры?
Как устроена клиент-серверная модель?
В чём суть разделения логики между клиентом и сервером?
Какие типы клиент-серверной архитектуры существуют (например: 2-tier, 3-tier)?
Какие плюсы и минусы у клиент-серверной архитектуры?
В чём отличие монолита от клиент-серверной архитектуры?
Как масштабируются монолитная и клиент-серверная архитектуры?
Как архитектура влияет на производительность системы?
Какие проблемы возникают при переходе от монолита к клиент-серверной архитектуре?
Как выбрать между монолитом и клиент-серверной архитектурой?
Приведите примеры продуктов с монолитной архитектурой.
Приведите примеры продуктов с клиент-серверной архитектурой.
Как клиент-серверная архитектура используется в веб-приложениях?
Какие протоколы используются в клиент-серверной архитектуре?
Как обеспечивается безопасность в клиент-серверной архитектуре?
Как монолитная архитектура связана с микросервисами?
Может ли клиент-серверная архитектура быть распределённой?
Какие технологии поддерживают клиент-серверную архитектуру?
Как архитектура влияет на CI/CD-процессы?
Как развивается клиент-серверная архитектура в условиях облачных вычислений?
Какие паттерны проектирования чаще всего используются в клиент-серверной архитектуре?
Как работают сессии и авторизация в клиент-серверной архитектуре?
Что такое REST API и как он связан с клиент-серверной архитектурой?
Какие инструменты используются для управления состоянием клиента и сервера?
Как тестировать клиент-серверное приложение?
Что такое программная архитектура?
В чём разница между архитектурой и проектированием?
Перечислите основные цели выбора архитектуры ПО.
Что такое монолитная архитектура?
Что такое клиент-серверная архитектура?
Как устроена монолитная архитектура? Опишите её компоненты.
Какие преимущества имеет монолитная архитектура?
Какие недостатки характерны для монолита?
В каких случаях применяется монолитная архитектура?
Как монолит влияет на скорость разработки и деплоя?
Что такое клиент и сервер в контексте архитектуры?
Как устроена клиент-серверная модель?
В чём суть разделения логики между клиентом и сервером?
Какие типы клиент-серверной архитектуры существуют (например: 2-tier, 3-tier)?
Какие плюсы и минусы у клиент-серверной архитектуры?
В чём отличие монолита от клиент-серверной архитектуры?
Как масштабируются монолитная и клиент-серверная архитектуры?
Как архитектура влияет на производительность системы?
Какие проблемы возникают при переходе от монолита к клиент-серверной архитектуре?
Как выбрать между монолитом и клиент-серверной архитектурой?
Приведите примеры продуктов с монолитной архитектурой.
Приведите примеры продуктов с клиент-серверной архитектурой.
Как клиент-серверная архитектура используется в веб-приложениях?
Какие протоколы используются в клиент-серверной архитектуре?
Как обеспечивается безопасность в клиент-серверной архитектуре?
Как монолитная архитектура связана с микросервисами?
Может ли клиент-серверная архитектура быть распределённой?
Какие технологии поддерживают клиент-серверную архитектуру?
Как архитектура влияет на CI/CD-процессы?
Как развивается клиент-серверная архитектура в условиях облачных вычислений?
Какие паттерны проектирования чаще всего используются в клиент-серверной архитектуре?
Как работают сессии и авторизация в клиент-серверной архитектуре?
Что такое REST API и как он связан с клиент-серверной архитектурой?
Какие инструменты используются для управления состоянием клиента и сервера?
Как тестировать клиент-серверное приложение?
Что такое многозвенная архитектура?
Какие уровни (звенья) могут присутствовать в многозвенной архитектуре?
В чём отличие двухзвенной архитектуры от трёхзвенной?
Какие компоненты участвуют в трёхзвенной архитектуре?
Какие преимущества даёт разделение на уровни?
Что такое толстый клиент? Приведите примеры.
Что такое тонкий клиент? Приведите примеры.

В чём основные различия между толстым и тонким клиентом?
Как распределяется логика между клиентом и сервером в зависимости от типа клиента?
Как выбор типа клиента влияет на производительность и масштабируемость?
Как выглядит архитектура с толстым клиентом?
Как выглядит архитектура с тонким клиентом?
В каких случаях целесообразно использовать толстый клиент?
В каких случаях предпочтителен тонкий клиент?
Какой тип клиента чаще используется в веб-приложениях?
Приведите примеры приложений с толстым клиентом.
Приведите примеры приложений с тонким клиентом.
Как толстые клиенты взаимодействуют с серверными компонентами?
Как тонкие клиенты получают данные из сервера?
Какие протоколы используются для взаимодействия между клиентом и сервером?
Какие паттерны проектирования применяются в многозвенных архитектурах?
Как работает MVC в контексте многозвенной архитектуры?
Что такое Presentation Tier, Business Tier, Data Tier?
Как реализуется разделение обязанностей в трёхзвенной архитектуре?
Как обеспечивается безопасность в многозвенной архитектуре?
Какие фреймворки поддерживают многозвенную архитектуру?
Как работают RIA (Rich Internet Applications)?
Как толстые клиенты реализуются в современных системах?
Какие инструменты используются для построения тонких клиентов?
Какие технологии обеспечивают работу клиент-серверного взаимодействия?
Как толстые клиенты связаны с desktop-приложениями?
Как тонкие клиенты связаны с веб-браузерами?
Что такое middle-tier и зачем он нужен?
Как можно оптимизировать передачу данных между клиентом и сервером?
Какие проблемы возникают при переходе от толстого клиента к тонкому?
Что такое сервис-ориентированная архитектура (SOA)?
Какие основные принципы характеризуют SOA?
Что такое сервис в контексте SOA?
Какие типы сервисов существуют?
В чём отличие SOA от монолитной архитектуры?
Что такое микросервисы?
Какие основные характеристики микросервисной архитектуры?
В чём сходство и различие между SOA и микросервисами?
Какие преимущества даёт микросервисная архитектура?
Какие сложности возникают при переходе к микросервисам?
Что такое REST API и как он используется в микросервисах?
Что такое gRPC и зачем он нужен?
Что такое API Gateway и как он работает?
Что такое Service Mesh (например, Istio)?
Что такое Discovery Service и как он участвует в работе микросервисов?
Как микросервисы общаются друг с другом?
Что такое синхронное и асинхронное взаимодействие?
Что такое message broker и какие из них популярны?
Какие паттерны используются для взаимодействия микросервисов (Event-driven, Saga и др.)?
Как обеспечивается транзакционность в распределённых системах?
Как микросервисы хранят данные?
Что такое bounded context в контексте микросервисов?
Как решается проблема дублирования данных?
Что такое CQRS и как оно применяется?
Что такое event sourcing и как оно связано с микросервисами?
Как микросервисы развертываются и масштабируются?
Что такое Docker и как он используется в микросервисах?
Что такое Kubernetes и зачем он нужен?
Как обеспечивается независимое развёртывание каждого сервиса?
Какие подходы к CI/CD используются в микросервисной архитектуре?
Как обеспечивается безопасность в микросервисной архитектуре?
Что такое OAuth 2.0, JWT и как они используются?
Как происходит аутентификация и авторизация в микросервисах?
Какие проблемы безопасности могут возникнуть в микросервисной архитектуре?
Как управлять конфигурациями и секретами в микросервисах?
Как выбрать между SOA и микросервисами?
Когда стоит использовать микросервисную архитектуру?
Какие ошибки чаще всего совершают при переходе на микросервисы?
Как тестировать микросервисы?
Как документировать API микросервисов?

Что такое Serverless Architecture и как она связана с микросервисами?
Как работают observability tools (Prometheus, Grafana, Jaeger)?
Какие инструменты помогают в управлении микросервисами?
Что такое resilience engineering и как он применяется?
Как микросервисы влияют на культуру DevOps?
Что такое облачные технологии?
В чём отличие между IaaS, PaaS и SaaS?
Какие основные модели доставки облачных сервисов?
Перечислите популярные облачные провайдеры.
Как облачные технологии влияют на разработку и эксплуатацию ПО?
Что такое контейнеризация?
В чём отличие виртуальной машины от контейнера?
Что такое Docker и зачем он нужен?
Как устроена архитектура Docker?
Что такое образ (image) и контейнер (container) в Docker?
Как создать Docker-образ?
Как запустить контейнер из образа?
Что такое Dockerfile и для чего он используется?
Как работает система слоёв (layers) в Docker?
Что такое Docker Hub и как он используется?
Что такое docker-compose и зачем он нужен?
Как использовать docker-compose.yml?
Как остановить и удалить контейнер?
Что такое volumes и bind mounts в Docker?
Как обеспечивается изоляция между контейнерами?
Что такое Kubernetes и зачем он нужен?
Как Kubernetes управляет контейнерами?
Что такое pod, deployment, service в Kubernetes?
Какие есть инструменты для управления Kubernetes?
Как Kubernetes взаимодействует с облачными провайдерами?
Как контейнеризация упрощает CI/CD-процессы?
Что такое CI/CD pipeline и как он связан с контейнерами?
Как автоматизировать сборку и деплой контейнеров?
Что такое immutable infrastructure?
Как контейнеры помогают в тестировании и отладке?
Как обеспечивается безопасность контейнеров?
Что такое scanning for vulnerabilities в контейнерах?
Как защитить образы Docker?
Что такое secrets management в контейнерной среде?
Как работать с сертификатами и ключами в контейнерах?
Что такое Serverless Architecture и как она связана с контейнерами?
Что такое Cloud Native и какие его компоненты?
Какие есть альтернативы Docker?
Что такое OCI (Open Container Initiative)?
Как работают sidecar-контейнеры?
Как оптимизировать размер Docker-образов?
Что такое multi-stage builds в Docker?
Как работать с microservices в контейнерной среде?
Что такое observability в контейнерной среде?
Как обеспечить high availability при работе с контейнерами?
Основы потоковой обработки данных
Что такое потоковая обработка данных (stream processing)?
В чём отличие пакетной и потоковой обработки?
Какие основные характеристики данных в потоковых системах?
Что такое event time, processing time и ingestion time?
Что такое windowing и зачем он нужен?
Что такое batch vs. stream processing?
Что такое event-driven architecture?
Что такое Kappa Architecture и как она отличается от Lambda?
Что такое real-time analytics и как он реализуется?
Какие уровни обработки событий могут быть в потоковой архитектуре?
Что такое message broker и какие из них популярны?
Что такое stream processor и какие из них существуют?
Что такое stateful processing и где он применяется?
Что такое checkpointing и state management?
Как работают sources и sinks в потоковой обработке?
Что такое Apache Kafka и как он используется в потоковых системах?
Что такое Apache Flink и как он отличается от Spark Streaming?

Что такое Apache Storm и где он применяется?
Что такое Apache Pulsar и чем он хорош?
Что такое Kafka Streams и когда его использовать?
Что такое event sourcing и как он связан с потоковой обработкой?
Что такое CQRS и как он применяется в потоковых системах?
Что такое event-driven microservices?
Что такое fan-out pattern?
Что такое aggregation over time windows?
Как обеспечивается отказоустойчивость в потоковых системах?
Что такое exactly-once semantics и как он достигается?
Как работает state checkpointing в Apache Flink?
Что такое backpressure и как он управляется?
Как системы обрабатывают пропущенные или дублированные события?
Как потоковые данные интегрируются с OLAP-системами?
Как происходит интеграция с базами данных (RDBMS, NoSQL)?
Как используются materialized views в потоковой обработке?
Как работать с real-time dashboards?
Как интегрировать потоковые данные с ML-моделями?
Как обеспечивается безопасность при передаче потоковых данных?
Как управлять доступом к потоковым системам?
Как логгируются и мониторятся потоки данных?
Как контролировать качество данных в потоках?
Как обеспечивается масштабируемость потоковых систем?
Что такое stream joins и как они работают?
Что такое event time ordering и как его обеспечить?
Какие есть подходы к тестированию потоковых приложений?
Что такое stream partitioning и зачем он нужен?
Как оптимизировать производительность потоковых вычислений?
Что такое интеграция программных систем?
Какие основные задачи решает интеграция?
В чём разница между внутренней и внешней интеграцией?
Почему возникает необходимость в интеграции различных систем?
Какие типичные проблемы возникают при интеграции?
Какие основные задачи стоят перед архитектором при проектировании интеграции?
Что такое обмен данными и как он реализуется?
Как обеспечивается синхронизация данных между системами?
Что такое маршрутизация сообщений и зачем она нужна?
Какие подходы используются для преобразования данных при интеграции?
Какие технические сложности могут возникнуть при интеграции?
Что такое проблема точечных связей (point-to-point integration)?
Как избежать дублирования логики в разных системах?
Что такое проблема синхронизации состояния?
Как обрабатываются ошибки и исключения в интеграционных системах?
Что такое стиль интеграции?
Перечислите основные стили интеграции.
Что такое точечная интеграция и её недостатки?
Что такое шины сообщений (message bus) и как они работают?
Что такое ESB (Enterprise Service Bus) и зачем он нужен?
Что такое ориентированная на сервисы архитектура (SOA) и как она используется для интеграции?
Что такое микросервисная архитектура и её роль в интеграции?
Что такое event-driven architecture и как она применяется?
Что такое publish-subscribe модель и где она используется?
Что такое API-ориентированный стиль интеграции?
Какие инструменты используются для реализации шин сообщений?
Что такое Apache Kafka и как он используется в интеграции?
Что такое RabbitMQ и когда его применять?
Что такое REST API и как он используется в интеграции?
Что такое gRPC и чем он лучше REST?
Что такое Adapter Pattern и как он применяется?
Что такое Facade Pattern и его роль в интеграции?
Что такое Bridge Pattern и зачем он нужен?
Что такое Mediator Pattern и как он помогает?
Что такое Message Queue Pattern и где он применяется?
Как организуется управление конфигурациями в интеграционных системах?
Как обеспечивается надёжность интеграционных процессов?
Как работает мониторинг и логирование в интеграционных системах?
Как тестировать интеграционные сценарии?
Как документировать интеграционные процессы?

Что такое Integration Testing и как его проводить?
Что такое orchestration vs. choreography в контексте интеграции?
Как использовать API Gateways в микросервисной архитектуре?
Что такое service mesh и как он влияет на интеграцию?
Какие есть современные тренды в интеграции систем?
Что такое корпоративная интеграция?
В чём заключается проблема точечных связей между системами?
Какие цели преследует корпоративная интеграция?
Что такое сервисная шина (ESB)?
В чём отличие ESB от простой шины сообщений?
Как устроена архитектура ESB?
Какие основные компоненты содержит ESB?
Как ESB обеспечивает маршрутизацию сообщений?
Что такое медиаторы в ESB и зачем они нужны?
Как ESB обрабатывает ошибки и исключения?
Какие преимущества даёт использование ESB?
Какие недостатки могут быть при внедрении ESB?
В каких случаях целесообразно использовать ESB?
Как ESB влияет на производительность системы?
Может ли ESB использоваться в микросервисной архитектуре?
Что такое паттерн интеграции?
Перечислите основные паттерны интеграции.
Что такое Message Router и как он работает?
Что такое Message Translator и зачем он нужен?
Что такое Content-Based Router и где он применяется?
Что такое Message Filter и как он используется?
Что такое Splitter и Aggregator?
Что такое Pipe and Filter и как он применяется?
Что такое Content Enricher и как он работает?
Что такое Message Broker и как он связан с ESB?
Что такое Request-Reply и как он реализуется?
Что такое Publish-Subscribe и где он используется?
Что такое Event Notification и как он применяется?
Что такое Correlation Identifier и зачем он нужен?
Что такое Dead Letter Channel и как он используется?
Какие популярные ESB-инструменты существуют (например, MuleSoft, Apache Camel)?
Что такое Apache Camel и какие паттерны он поддерживает?
Что такое WSO2 ESB и его особенности?
Как ESB интегрируется с REST API и SOAP-сервисами?
Какие языки маршрутизации используются в ESB (например, DataWeave, XPath)?
Как ESB обеспечивает управление конфигурациями?
Как работает мониторинг и логирование в ESB?
Как обеспечивается безопасность в ESB?
Как ESB обрабатывает нагрузку и масштабируется?
Как тестировать интеграционные сценарии в ESB?
В чём разница между ESB и API Gateway?
Как ESB взаимодействует с микросервисами?
Что такое lightweight integration patterns и их роль?
Какие современные альтернативы ESB существуют?
Какие тренды в интеграции сейчас наиболее актуальны?
Что такое качество архитектуры программной системы?
Какие основные критерии качества архитектуры выделяют в промышленных системах?
В чём разница между функциональными и нефункциональными требованиями?
Почему важны нефункциональные требования при проектировании архитектуры?
Как качество архитектуры влияет на жизненный цикл продукта?
Что такое надёжность архитектуры?
Как обеспечивается работоспособность системы при сбоях?
Что такое высокая доступность (high availability) и как её достигнуть?
Что такое fault tolerance и как он реализуется?
Какие методы используются для обеспечения failover?
Что такое масштабируемость архитектуры?
В чём разница между вертикальной и горизонтальной масштабировостью?
Какие архитектурные решения способствуют масштабируемости?
Что такое load balancing и как он применяется?
Как обеспечить распределённую обработку нагрузки?
Что такое производительность архитектуры?
Какие метрики используются для оценки производительности?
Как оптимизировать время отклика системы?

Что такое latency и как его уменьшить?
Как обеспечить эффективное использование ресурсов?
Что такое поддерживаемость архитектуры?
Какие архитектурные паттерны способствуют поддержке?
Что такое maintainability и как её повысить?
Как документация влияет на поддерживаемость?
Что такое observability и зачем она нужна?
Как архитектура влияет на тестируемость?
Что такое testability и как её обеспечить?
Какие подходы к тестированию архитектуры существуют?
Что такое unit tests, integration tests, end-to-end tests?
Как архитектура влияет на автоматизацию тестирования?
Что такое гибкость архитектуры?
Какие архитектурные шаблоны способствуют гибкости?
Что такое адаптивная архитектура?
Как обеспечить обратную совместимость?
Как архитектура влияет на возможность модификации и расширения?
Что такое безопасность архитектуры?
Какие аспекты безопасности нужно учитывать при проектировании?
Как обеспечить аутентификацию и авторизацию?
Что такое защита от DDoS-атак и как её организовать?
Как архитектура влияет на шифрование данных и передачу информации?
Что такое простота архитектуры?
Как избежать избыточности в проектировании?
Что такое KISS и YAGNI и как они связаны с качеством?
Как обеспечить единообразие решений в проекте?
Как архитектура влияет на понятность кода и документов?
Что такое архитектурный стиль и как он влияет на качество?
Какие инструменты используются для анализа архитектуры (например, SonarQube, ArchUnit)?
Что такое архитектурные эволюционные метрики?
Как оценивать архитектуру при переходе от монолита к микросервисам?
Какие есть методики оценки качества архитектуры (например, ATAM, Architecture Tradeoff Analysis Method)?

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

6.1. Рекомендуемая литература

6.1.1. Основная литература

№	Авторы, составители	Заглавие	Издательство, год	Количество/ название ЭБС
Л1.1		Вестник Российской правовой академии	, 2005,	2072-9936, http://www.iprbookshop.ru/3325.html
Л1.2	Болодурина И. П., Волкова Т. В.	Проектирование компонентов распределенных информационных систем : учебное пособие	Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2012, 215 с.	978-5-4417-0077-1, http://www.iprbookshop.ru/30122.html
Л1.3	Зиангирова Л. Ф.	Технологии облачных вычислений : учебное пособие	Саратов: Вузовское образование, 2016, 300 с.	2227-8397, http://www.iprbookshop.ru/41948.html
Л1.4	Зубкова Т. М.	Технология разработки программного обеспечения : учебное пособие	Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017, 469 с.	978-5-7410-1785-2, http://www.iprbookshop.ru/78846.html

№	Авторы, составители	Заглавие	Издательство, год	Количество/ название ЭБС
Л1.5	Фаулер М.	Архитектура корпоративных программных приложений : Пер.с англ.	М.:СПб.:Киев: Вильямс, 2004, 544с.	5-8459-0579-6, 1
Л1.6	Альтман Е. А., Александров А. В., Васеева Т. В.	Система контроля версий GIT : учебно-методическое пособие к выполнению лабораторных работ	Омск: ОмГУПС, 2021, 26 с.	, https://e.lanbook.com/book/190155
Л1.7	Баланов А. Н.	Построение микросервисной архитектуры и разработка высоконагруженных приложений : учебное пособие для вузов	Санкт-Петербург: Лань, 2024, 244 с.	978-5-507-48747-9, https://e.lanbook.com/book/394538
Л1.8	Федькова, Н. А.	Современные технологии разработки программного обеспечения : методическое пособие	Брянск: Брянский государственный аграрный университет, 2022, 58 с.	2227-8397, https://www.iprbookshop.ru/138519.html

6.1.2. Дополнительная литература

№	Авторы, составители	Заглавие	Издательство, год	Количество/ название ЭБС
Л2.1	Карпович, Е. Е.	Методы тестирования и отладки программного обеспечения : учебник	Москва: Издательский Дом МИСиС, 2020, 136 с.	978-5-907226-64-7, http://www.iprbookshop.ru/106722.html

6.1.3. Методические разработки

№	Авторы, составители	Заглавие	Издательство, год	Количество/ название ЭБС
Л3.1	Рощин П. Г.	Командная разработка программного обеспечения с помощью системы контроля версий Git: Конспект лекций : учебное пособие	Москва: НИЯУ МИФИ, 2022, 108 с.	978-5-7262-2846-4, https://e.lanbook.com/book/355550

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"

Э1	
Э2	

6.3 Перечень программного обеспечения и информационных справочных систем

6.3.1 Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства

Наименование	Описание
Pyton	Свободное ПО
Операционная система Windows	Коммерческая лицензия
SumatraPDF	Свободное ПО
Notepad++	Свободное ПО
Firefox	Свободное ПО
Far Manager 3	Свободное ПО
PyCharm Community	Свободное ПО

6.3.2 Перечень информационных справочных систем

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

1	128 учебно-административный корпус. учебная аудитория для прове-дения учебных занятий Специализированная мебель (24 посадочных мест), магнитно-маркерная доска, мультимедиа проектор (Ben-Q), 1 экран, звуковые колонки. ПК: AMD A10-6700/8Gb – 10 шт., AMD A10 PRO-7800B/8Gb – 4 шт., Intel i3-2120/8Gb – 1 шт., Intel 2 Duo E7200/6Gb – 1 шт. Возможность подключения к сети Интернет и обеспечением доступа в электронную информационно-образовательную среду РГРТУ
2	155 учебно-административный корпус. учебная аудитория для проведения учебных занятий Специализированная мебель (24 посадочных мест), магнитно-маркерная доска, интерактивная доска, мультимедиа проектор (Toshiba), звуковые колонки. ПК: Intel i5-3470/8Gb – 12 шт., Intel i5-2400/8Gb – 2 шт., Intel 2 Duo E7200/4Gb – 2 шт. Возможность подключения к сети Интернет и обеспечением доступа в электронную информационно-бразовательную среду РГРТУ
3	414 учебно-административный корпус. Помещение для самостоятельной работы Специализированная мебель (40 посадочных мест), магнитно-маркерная доска, экран. Мультимедийный проектор (NEC AOC 2050W) ПК: Intel Pentium G620/4Gb – 13 шт Возможность подключения к сети Интернет и обеспечением доступа в электронную информационно-образовательную среду РГРТУ

8. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Методические обеспечение дисциплины приведено в приложении к рабочей программе дисциплины в файлах:
- Практика kafka-python.pdf
- Задание на практику по PyPipeline.pdf
- Практика по PyPipeline.pdf
- PyPipeline-ESB_EIP.pdf
- Операторы_Apache-Flink.pdf
- Flink_Практика.pdf

Оператор ЭДО ООО "Компания "Тензор"

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ

ПОДПИСАНО ЗАВЕДУЮЩИМ КАФЕДРЫ	**ФГБОУ ВО "РГРТУ", РГРТУ,** Корячко Вячеслав Петрович, Заведующий кафедрой САПР	**07.10.25** 14:09 (MSK)	Простая подпись
ПОДПИСАНО ЗАВЕДУЮЩИМ ВЫПУСКАЮЩЕЙ КАФЕДРЫ	**ФГБОУ ВО "РГРТУ", РГРТУ,** Корячко Вячеслав Петрович, Заведующий кафедрой САПР	**07.10.25** 14:10 (MSK)	Простая подпись